TITLE:

Binary Splitting Algorithmによる高精度計算 (数式処理における理論と応用の研究)

AUTHOR(S):

平山, 弘

CITATION:

平山, 弘. Binary Splitting Algorithmによる高精度計算 (数式処理における理論と応用の研究). 数理解析研究所講究録 2001, 1199: 160-166

ISSUE DATE:

2001-04

URL:

http://hdl.handle.net/2433/64918

RIGHT:

京都大学学術情報リポジトリ
KURENAI 紅
Kyoto University Research Information Repository

数理解析研究所講究録
1199巻 2001年 160-166

Binary Splitting Algorithmによる高精度計算

神奈川工科大学　平山　弘(Hiroshi HIRAYAMA) *

1　はじめに

級数の高精度計算を有理数を使って計算することによって、計算速度を上げる方法が多くの人によって使われている。この方法を使って、1989年にChudnovsky兄弟による円周率10億桁の計算が行われた。しかしながら、彼らは、この計算方法の詳細を発表していない。

この方法は1998年から1999年にかけて、右田等[11][12]や後等[15]によって、級数計算への適用の再現が行われている。

この計算法の起源を調べると1997年にすでにB.Haible and T.Papanikolaou [3]によって公表されていることがわかる。この論文では、トーナメント法とか縮約法とか呼ばれているこれらの計算方法をBinary Splitting Algorithm（以降BSA法と略す）と呼んでいる。

この論文を読むとBSA法は、1976年にR. P. Brent、1987年にBorwein兄弟によって公表され、Chudnovsky兄弟による円周率10億桁の計算が行われた1989年当時でもよく知られた方法であったことがわかる。

B.Haible等の論文では、次の形の無限級数

$$S=\sum_{k=0}^{\infty}\left(\frac{a_k}{b_k}\frac{\prod_{j=0}^{k}p_j}{\prod_{j=0}^{k}q_j}\right) \tag{1}$$

だけでなく、さらに一般的な無限級数

$$U=\sum_{k=0}^{\infty}\left(\frac{a_k}{b_k}\left(\sum_{j=0}^{\infty}\frac{c_j}{d_j}\right)\frac{\prod_{j=0}^{k}p_j}{\prod_{j=0}^{k}q_j}\right) \tag{2}$$

の高速化も容易に行えることを述べている。ここで、a_k、b_k、c_k、d_k、p_k、q_kは、$O(\log k)$の桁数の整数で、通常kの定数係数の多項式である。

BSA法は、l、m、n $(l<m<n)$を正の整数としたとき、次のような手順になる。まず、

$$S_{l,m-1}=\sum_{k=l}^{m-1}\left(\frac{a_k}{b_k}\frac{\prod_{j=0}^{k}p_j}{\prod_{j=0}^{k}q_j}\right) \tag{3}$$

*hirayama@sd.kanagawa-it.ac.jp

$$P_{l,m-1} = \prod_{j=l}^{m-1} p_j \tag{4}$$

$$Q_{l,m-1} = \prod_{j=l}^{m-1} q_j \tag{5}$$

$$B_{l,m-1} = \prod_{j=l}^{m-1} b_j \tag{6}$$

と記する。これを使って

$$T_{l,m-1} = B_{l,m-1} Q_{l,m-1} S_{l,m-1} \tag{7}$$

を定義する。このとき、次のような漸化式

$$P_{l,n-1} = P_{l,m-1} P_{m,n-1} \tag{8}$$

$$Q_{l,n-1} = Q_{l,m-1} Q_{m,n-1} \tag{9}$$

$$B_{l,n-1} = B_{l,m-1} B_{m,n-1} \tag{10}$$

$$\begin{aligned} T_{l,n-1} &= B_{m,n-1} Q_{m,n-1} T_{l,m-1} \\ &+ B_{l,m-1} P_{l,m-1} T_{m,n-1} \end{aligned} \tag{11}$$

が成り立つ。この関係を何回も利用して、$l = 0$ から大きな n までの $Q_{0,n-1}$、$B_{0,n-1}$、$T_{0,n-1}$ の値を求め

$$S_{0,n-1} = \frac{T_{0,n-1}}{B_{0,n-1} Q_{0,n-1}} \tag{12}$$

として、S の近似値を計算することができる。右田等 [11][12] や後等 [15] による級数計算への適用は、この S を計算することに対応する。

さらに

$$U_{l,m-1} = \sum_{k=l}^{m-1} \left(\frac{a_k}{b_k} \Big(\sum_{j=l}^{k} \frac{c_j}{d_j} \Big) \frac{\prod_{j=0}^{k} p_j}{\prod_{j=0}^{k} q_j} \right) \tag{13}$$

$$D_{l,m-1} = \prod_{j=l}^{m-1} d_j \tag{14}$$

$$C_{l,m-1} = D_{l,m-1} \sum_{j=l}^{m-1} \frac{c_j}{d_j} \tag{15}$$

$$V_{l,m-1} = D_{l,m-1} B_{l,m-1} Q_{l,m-1} U_{l,m-1} \tag{16}$$

と記するとき、次の漸化式

$$D_{l,n-1} = D_{l,m-1} D_{m,n-1} \tag{17}$$

$$\begin{aligned} C_{l,n-1} &= C_{l,m-1}D_{m,n-1} \\ &+ C_{m,n-1}D_{l,m-1} \end{aligned} \tag{18}$$

$$\begin{aligned} V_{l,n-1} &= D_{m,n-1}B_{m,n-1}Q_{m,n-1}V_{l,m-1} \\ &+D_{m,n-1}C_{l,m-1}B_{l,m-1}P_{l,m-1}T_{m,n-1} \end{aligned} \tag{19}$$

が成り立つ。この関係式から、前の S と同様に、$l=0$ から大きな n までの $D_{0,n-1}$、$Q_{0,n-1}$、$B_{0,n-1}$、$V_{0,n-1}$ を求める。この値から

$$U_{0,n-1} = \frac{V_{0,n-1}}{D_{0,n-1}B_{0,n-1}Q_{0,n-1}} \tag{20}$$

を計算し、U の近似値を計算することができる。この計算法は、(1) および (2) 式において、a_k、b_k、c_k、d_k、p_k、q_k が小さな整数で、級数の値を高精度で計算する場合に効果的である。

このような一般化によって、Bessel 関数 $K_0(x)$ が BSA 法によって計算できることを意味する。この Bessel 関数を使って R.P.Brent 等は Euler 定数を計算する方法 [14] を提案しているが、計算例では BSA 法でない通常の計算方法によるものである。

効率的である理由は二つある。第一番目の理由は、長い桁数の数値が途中で出現しないため小さな桁数でかつ厳密な値で計算を進めることができるためである。(15) 式は、除数を因数として持つ $D_{l,m-1}$ を掛けるため整数値になるので、この除算のために長い桁数の数値が現れることはない。第二番目の理由は、数値の桁数が大きくなった場合、大きな桁数の数値の乗算に FFT を用いた高速乗算法を適用できるためである。

2 BSA 法による基数変換

基数変換については、1999 年の講演とその講究 [8] や後等 [16] によって BSA 法によって高速に計算できる可能性が述べられている。現在の多くの計算機が内部表現として 2 進数を利用している。多倍長数値表現として 2 進数表示を使うと、メモリー効率だけでなく計算速度もかなり速くなる。円周率の高精度計算等の特殊な計算を除けは通常 2 進数を使う場合が多い。このため、2 進数を 10 進数への変換は、基本的で実際にもよく行われる計算で非常に重要である。実際にどの程度高速計算できるかを以下で調べた。

実数 a を 2 進数で表現すると、

$$a = \sum_{k=0}^{\infty} 2^{N-k} \tag{21}$$

と表現できる。これは、次のような形式に変形できる。

$$\begin{aligned} a &= a_0 2^N + a_1 2^{N-1} + a_2 2^{N-2} + \cdots \\ &= 2^N(a_0 + \tfrac{1}{2}(a_1 + \tfrac{1}{2}(a_2 + \tfrac{1}{2}(a_3 + \cdots \end{aligned} \tag{22}$$

この式を、基数 r の式に一般化すると、

$$a = \sum_{k=0}^{\infty} r^{N-k} \tag{23}$$

となる。(1) の形式に変形すると

$$\begin{aligned} a &= a_0 r^N + a_1 r^{N-1} + a_2 r^{N-2} + \cdots \\ &= r^N (a_0 + \tfrac{1}{r}(a_1 + \tfrac{1}{r}(a_2 + \tfrac{1}{r}(a_3 + \cdots \end{aligned} \tag{24}$$

この計算を、10 進数表現の多倍長演算プログラムで計算すれば、10 進数に変換できる。この時の計算量は、精度を p とすれば $O(p(\log p)^3)$ の計算量で計算できる。実際の計算では、(24) の基数 r は、2^{16} とか 2^{32} になる。

(24) の級数部分を行列の乗算形式で書くと、

$$\begin{pmatrix} 1 & P_n \\ 0 & R_n \end{pmatrix} = \begin{pmatrix} 1 & a_0 \\ 0 & r \end{pmatrix} \begin{pmatrix} 1 & a_1 \\ 0 & r \end{pmatrix} \begin{pmatrix} 1 & a_2 \\ 0 & r \end{pmatrix} \cdots \begin{pmatrix} 1 & a_n \\ 0 & r \end{pmatrix} \tag{25}$$

となる。この式を計算し、$a = r^N \frac{P_n}{R_n}$ として、a が求められる。

逆に、基数 r を 10 にして、2 進数表現を使う多倍長演算プログラムで計算すれば、2 進数に変換することになる。この時の計算量も、2 進数 10 進数の変換と同様に、$O(p(\log p)^3)$ の計算量で計算できる。

実際に、この計算法を利用してすれば、以下のような時間となった。使用した計算機は Pentium Ⅲ 933MHz、Windows 2000、C++コンパイラとして、CBuilder　Ver.5 を利用した。$\sqrt{3}$ を計算し、それを 10 進数に変換した。正確には、2^{32} 進数の数値を 10^4 進数に変換した。

以下にその結果を表 1 に示す。

表 1　2 進数を 10 進数に変換するための時間

計算桁数	従来の方法 (msec)	本方法 (msec)	$\sqrt{3}$ の平方根の計算 (msec)
50	0.06	0.09	0.08
100	0.14	0.17	0.09
200	0.29	0.29	0.17
500	1.23	0.98	0.71
1000	4.37	2.82	2.19
2000	14.85	8.13	5.32
5000	72.66	32.96	15.47
10000	250.0	73.5	34.4
20000	917.2	164.0	73.4
50000	5,297	407	188
100000	20,781	1,125	500
200000	82,360	3,156	1,422

50 桁の場合 10000 回、500 桁以下の場合 1000 回、5000 桁以下の場合 100 回、20000 桁以下の場合 10 回ループさせ、時間を測定し、そのループ回数で割った値である。

上の結果を見ると BSA 法を使った方法は、500 桁程度以上の精度では、従来の方法より高速であることがわかる。従来の方法は、100 桁程度以下の計算で高速であることがわかる。２進数を 10 進数に変換する演算は、50 桁程度までの精度では、平方根の計算より高速に行うことができるが、それ以上の精度の計算では、ここで提案した高速アルゴリズムを使っても遅くなることがわかる。

3 指数関数の計算

RSA 法は、桁数の短い数値の関数の計算だけでなく、高精度の数値の関数計算にも適用できる。高精度数 x を桁数の少ない上位桁部分 x_0 とそれ以外の部分 x_1 に分ける。すなわち、

$$e^x = e^{x_0} e^{x_1} \quad x = x_0 + x_1 \tag{26}$$

として計算する。e^{x_0} は、Taylor 展開で計算するとき、x_0 の桁数が小さいことから、高速に計算できると期待できる。この計算には、BSA 法が使えるので、さらに高速に計算できる。

e^{x_1} の部分は、x_1 が非常に小さな数値になるので、級数は速く収束するため高速に計算できる。

数値例として、$e^{\pi} = 23.140692632\cdots$ を計算精度 8000 桁で計算する。Brent の高精度計算ルーチンで使われている引数を２分割し、小さな引き数にし、その値の関数を Taylor 展開式で計算し、２乗を２分割した回数だけ行い、元の関数値を計算する。すなわち

$$e^{\pi} = \left(e^{\frac{\pi}{2^{202}}}\right)^{2^{202}} \tag{27}$$

を計算する。このとき、Pentium II 450MHz で 2.36 秒に時間がかかった。これを

$$e^{\pi} = \left(e^{\frac{\pi}{2^{20}}}\right)^{2^{20}} \tag{28}$$

を上位 96 桁と下位 7904 桁に分割し、上位の桁の計算に BSA 法を利用した。上位桁の計算は、254 次、下位は、64 次の Taylor 展開式を使用した。このときの計算時間は 1.42 秒であった。この値は Brent の高精度ルーチンの計算法より約 40 パーセント高速であった。

このような使い方をすれば、多くの関数が BSA 法で高速に計算できる可能性がある。

このような高精度計算方法については、後等 [16] でも述べられている。

4 まとめ

- BSA のアルゴリズムを使うと級数の高精度計算を高速化することができる。

- 関数の計算だけでなく基数変換にもこのアルゴリズムを使うことができる。
- 桁数の小さな数値で構成された級数だけでなく、フル精度の計算も高速に計算できる。

参 考 文 献

[1] Bergland G. D.: A Radix-Eight Fast Fourier Transform Subroutine for Real-Valued Series, IEEE Trans. A. E., AU17.2, pp. 138–144

[2] Henrici P.: Applied and Computational Complex Analysis, Vol. 3, Chap. 13, John Wiley & Sons, New York(1986)

[3] Haible B., Papanikolaou T.: Fast multiprecision evaluation of series of rational numbers, Technical Report No. TI-7/97, Darmstadt University of Technology, http://www.informatik.tu-darmstadt.de/TI/Mitarbeiter/papanik/(1997)

[4] 平山 弘, 浮川　直章: 連分数の高速評価法、情報処理学会研究報告、vol.99, No. 74, pp. 31–36(1998)

[5] 平山 弘: C++言語による高精度計算パッケージの開発, 日本応用数理学会, Vol. 5, No.3, pp. 123–134 (1995)

[6] 平山 弘: 多倍長計算プログラムパッケージ MPPACK の利用の手引き, 東大計算センター (1992)

[7] 平山　弘: 連分数の多倍長精度高速計算法、情報処理学会論文誌、Vol 41. No.SIG 8, pp. 85–91(2000)

[8] 平山　弘: 分割統治法による多倍長演算の高速化, 京都大学数理解析研究所講究録, Vol. 1138, pp. 247–255 (2000)

[9] Lehmer D.H.: Euclid's Algorithm for Large Numbers, Amer. Math. Monthly, Vol. 45, pp. 227–233(1938)

[10] Lorentzen L., Waadeland H.: Continued Fractions with Applications, North-Holland, Amsterdam(1992)

[11] 右田剛史、天野晃、浅田尚紀、藤野清次: 級数の集約による多倍長数の計算法とπの計算への応用、情報処理学会研究報告、vol.98, No. 74, pp. 31–36(1998)

[12] 右田剛史、天野晃、浅田尚紀、藤野清次: 級数の再帰的集約による多倍長数の計算法とπの計算への応用、情報処理学会論文誌、vol.40, No. 12, pp. 4193–4200(1999)

[13] 森、名取、鳥居: 数値計算（岩波講座情報科学１８）, ５章, 岩波書店 (1982)

[14] Richard P. Brent, and Edwin M. McMillan: Some new algorithms for high-precision computation of euler's constant. Mathematics of Computation vol. 34, pp. 305-312 (1980)

[15] 後保範、金田康正、高橋大介、無限級数に基づく多数桁計算の演算量削減を実現する分割有理数化法、京都大学数理解析研究所講究録 Vol.1084, pp. 60–71 (1999)

[16] 後保範、金田康正、高橋大介、級数に基づく多数桁計算の演算量削減を実現する分割有理数化法、情報処理学会論文誌 Vol.41, No. 6, pp. 1811–1819 (2000)